Improving Consistency Models with Generator-Induced Coupling

Thibaut Issenhuth[1] **Ludovic Dos Santos**[1] **Jean-Yves Franceschi**[1] **Alain Rakotomamonjy**[1]

Abstract

Consistency models are promising generative models as they distill the multi-step sampling of score-based diffusion in a single forward pass of a neural network. Without access to sampling trajectories of a pre-trained diffusion model, consistency training relies on proxy trajectories built on an independent coupling between the noise and data distributions. Refining this coupling is a key area of improvement to make it more adapted to the task and reduce the resulting randomness in the training process. In this work, we introduce a novel coupling associating the input noisy data with their generated output from the consistency model itself, as a proxy to the inaccessible diffusion flow output. Our affordable approach exploits the inherent capacity of consistency models to compute the transport map in a single step. We provide intuition and empirical evidence of the relevance of our generator-induced coupling (GC), which brings consistency training closer to score distillation. Consequently, our method not only accelerates consistency training convergence by significant amounts but also enhances the resulting performance.

1. Introduction

Diffusion and score-based models (Ho et al., 2020; Song et al., 2021; Karras et al., 2022) have emerged as state-of-the-art generative models for image generation. Since they are costly to use at inference time, as they require several neural function evaluations, many distillation techniques have been explored (Salimans and Ho, 2022; Meng et al., 2023; Sauer et al., 2023). A most remarkable approach is the one coined as *consistency models* (Song et al., 2023; Song and Dhariwal, 2024). Consistency models lead to high-quality one-step generators, that can be trained either by distillation of a score-based model, or as standalone generative models. In this paper, we study consistency models trained without a pre-trained score model.

At the core of the training of consistency models, an independent coupling (IC) between data and noise distribution is used. Indeed, when training a consistency model without a pre-trained score model, any point from the data distribution is matched to any point from the noise distribution in order to construct trajectories. However, Pooladian et al. (2023) identified those as a potential issue in flow matching models (Lipman et al., 2023). Notably, the authors demonstrated that IC leads to high variance of the gradient estimator of such generative models. They resort to batch coupling with optimal transport (OT) tools and prove that it improves performance. Dou et al. (2024) show that this approach can successfully be adopted in consistency models, and we actually confirm this in our experiments. However, discrete OT has its flaws. Most notably, it converges slowly to the true Wasserstein distance in high dimension, and increasing the size of the coupling sets leads to increased running time, since solvers (*e.g.* Hungarian matching algorithm) typically have a cubic complexity.

Consistency models map any point on a given reverse diffusion trajectory to the initial data point by minimizing the distance between the outputs of adjacent points on the trajectory. Training a consistency model in such a setting relies on constructing trajectories from an IC between data and noise distributions, and selecting pairs of intermediate points from each single trajectory (Song et al., 2023). Drawing inspiration from Pooladian et al. (2023), we aim at constructing better couplings between data and noise distributions. Instead of using discrete OT tools, we take another direction and propose to use the generator itself to construct trajectories. From an intermediate point computed from an IC, we let the consistency model predict the corresponding endpoint, supposedly close to the data distribution. This predicted endpoint is used to construct a new coupling, from which two points are selected to train the consistency model itself. Our approach is illustrated in Figure 1. Our claim is that it provides couplings that are better aligned to the actual diffusion updates and thus approximate better the true loss function of a consistency model. As shown in our experimental section, this procedure results in faster convergence and often leads to better performance of the resulting model.

[1]Criteo AI Lab, Paris, France. Correspondence to: Thibaut Issenhuth <t.issenhuth@criteo.com>.

Accepted by the Structured Probabilistic Inference & Generative Modeling workshop of ICML 2024, Vienna, Austria. Copyright 2024 by the author(s).

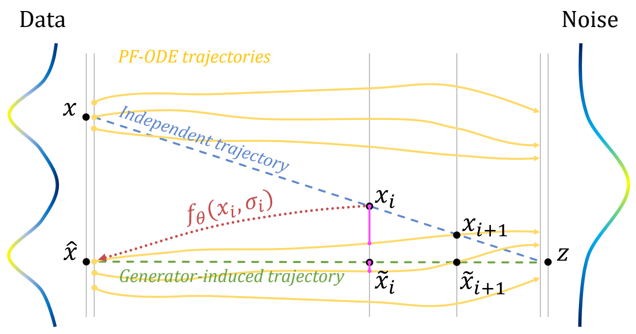


Figure 1. We leverage the current consistency model predictions to compute a new coupling used for its own training. Using the original independent coupling (IC) (x, z) and intermediate points (x_i, x_{i+1}) we derive a generator-induced coupling (GC) $(f_\theta(x_i, \sigma_i), z)$ and intermediate points $(\tilde{x}_i, \tilde{x}_{i+1})$. In purple, we illustrate that using generator-induced trajectories instead of the usual independent trajectories results in pairs of points closer to the ideal PF-ODE update of the corresponding point at $i + 1$.

Let us summarize our contributions below:

- We propose a novel type of coupling, denoted *generator-induced coupling* (GC), that allows to draw *generator-induced trajectories* used to construct pairs of points for the training of consistency model. We carefully identify and select the main design components of our method through experiments.
- We provide insights into the advantages of this coupling on both synthetic and image datasets. Most notably, we show that trajectories drawn from GC are closer to ideal trajectories defined by a pre-trained score model than standard trajectories drawn from IC.
- We experimentally demonstrate the benefits of this coupling in image generation benchmarks. When combined with standard trajectories from IC, GC leads to faster convergence and often improves the performance of the generative model.

2. Preliminaries

Notation. We consider an empirical data distribution $p_\star$ and noise distribution p_z (*e.g.* Gaussian) defined on $\mathbb{R}^d$. We note q a joint distribution of samples from $p_\star$ and p_z. We use a distance function $\mathcal{D}$ to measure the distance between two points from $\mathbb{R}^d$. sg denotes the stop-gradient operator.

In consistency models, we consider a discrete formulation of diffusion models with N intermediate timesteps. A vector from an intermediate timestep can be sampled as $x_i = \alpha_i x + \sigma_i z$, where $x \sim p_\star$, $z \sim p_z$, and (α_i) (resp. monotonous (σ_i)) are data (resp. noise) schedules. In the commonly used EDM process (Karras et al., 2022), $\alpha_i = 1$ and can thus be omitted. We denote $(p_{\sigma_i})_{i \in [0,N]}$ the corresponding intermediate noisy distributions between $p_{\sigma_0} \approx p_\star$ and $p_{\sigma_N} \approx p_z$, such that $x_i \sim p_{\sigma_i}$.

2.1. Score-Based Diffusion Models

Score-based diffusion models (Ho et al., 2020; Song et al., 2021) can generate data from noise via a multi-step process consisting in numerically solving either a stochastic differential equation (SDE), or an equivalent ordinary differential equation (ODE) called the probability flow ODE (PF-ODE). Although SDE solvers generally exhibit superior sampling quality, the PF-ODE has desirable properties. Notably, it defines a deterministic map from noise to data. In this context, the diffusion process based on the EDM formulation (Karras et al., 2022) with $\alpha_i = 1$ defines it as follows:

$$\mathrm{d}x = -\sigma \nabla_x \log p_\sigma(x)\, \mathrm{d}\sigma, \tag{1}$$

where $\nabla_x \log p_\sigma$, *a.k.a.* the score function, can be approximated with a neural network $s_\Phi(x, \sigma)$ (Vincent, 2011; Song and Ermon, 2019).

2.2. Consistency Models

Numerically solving the PF-ODE is costly because of multiple expensive evaluations of the score function. To alleviate this issue, Song et al. (2023) proposed to leverage the deterministic property of the PF-ODE in a *consistency model* f_θ, learning the output map of the PF-ODE, such that:

$$f_\theta(x_i, \sigma_i) = x_0, \tag{2}$$

for all $(x_i, \sigma_i) \in \mathbb{R}^D \times [\sigma_0, \sigma_N]$ that belong to the trajectory of the PF-ODE ending at (x_0, σ_0).

Equation (2) is equivalent to *(i)* enforcing the boundary condition $f_\theta(x_0, \sigma_0) = x_0$ and *(ii)* ensuring that f_θ has the same output for any two samples of a single PF-ODE trajectory—the consistency property. *(i)* is naturally satisfied by the following model parametrization:

$$f_\theta(x_i, \sigma_i) = c_{\text{skip}}(i) x_i + c_{\text{out}}(i) F_\theta(x_i, \sigma_i) \tag{3}$$

where $c_{\text{skip}}(0) = 1$, $c_{\text{out}}(0) = 0$, and F_θ is a neural network. *(ii)* is achieved by minimizing the distance between the outputs of two same-trajectory consecutive samples with the consistency loss:

$$\begin{aligned} L_{\text{consistency-distillation}}(\theta) = \mathbb{E}_{i \sim \mathcal{U}([0,N]), x_{i+1} \sim p_{\sigma_{i+1}}} \Big[\\ \mathcal{D}\Big(\operatorname{sg}(f_\theta(x_i, \sigma_i)), f_\theta(x_{i+1}, \sigma_{i+1})\Big)\Big], \end{aligned} \tag{4}$$

where x_i is computed by discretizing the PF-ODE of Equation (1) with the Euler scheme as follows:

$$x_i = x_{i+1} - (\sigma_i - \sigma_{i+1}) \sigma_{i+1} \nabla_{x_{i+1}} \log p_{\sigma_{i+1}}(x_{i+1}). \tag{5}$$

The latter loss can be directly used to distill a score model $s_\Phi(x_{i+1}, \sigma_{i+1}) \approx \nabla \log p_{\sigma_{i+1}}(x_{i+1})$ into f_θ. In the case of consistency training, Song et al. (2023) circumvent the lack of score function by noticing that $\nabla \log p_{\sigma_{i+1}}(x_{i+1}) = \mathbb{E}\left[\frac{x_{i+1}-x}{\sigma_{i+1}^2} | x_{i+1}\right]$. This results in selecting pairs of points $x_i = x + \sigma_i z$ and $x_{i+1} = x + \sigma_{i+1} z$ with the same *independent* $(x, z) \sim q_{\mathrm{I}} = p_\star(x) p_z(z)$:

$$L_{\text{consistency}}(\theta) = \mathbb{E}_{(x,z)\sim q_{\mathrm{I}}, x_i = x+\sigma_i z, x_{i+1}=x+\sigma_{i+1} z}\Big[\mathcal{D}\Big(\mathrm{sg}(f_\theta(x_i, \sigma_i)), f_\theta(x_{i+1}, \sigma_{i+1})\Big)\Big], \tag{6}$$

which converges to the distillation loss when $N \to \infty$.

3. Data-Noise Coupling

As seen in Section 2.2, consistency models compute vectors from intermediate timesteps through an IC $q_{\mathrm{I}} = p_\star(x) p_z(z)$ of data and noise, in a similar fashion to flow matching (Lipman et al., 2023). This IC takes root in score-based diffusion models which have inspired these more recent approaches (Kingma and Gao, 2024). Indeed, approximating the score function with a neural network entails sampling data points perturbed by a Gaussian kernel (Vincent, 2011). Accordingly, the diffusion-equivalent Denoising Diffusion Probabilistic Model (DDPM, Ho et al., 2020) initiates by independently sampling $x \sim p_\star$ and $z \sim p_z$. However, recent work has shown the benefits of choosing more adapted couplings in flow matching then in consistency models, thereby going beyond the original inspiration from diffusion models.

Beyond Independent Coupling (IC). The reliance on IC in consistency and flow models is increasingly recognized as a limiting factor. Recent advancements (Liu et al., 2023; Pooladian et al., 2023) suggest that improved coupling mechanisms could enhance both training efficiency and the quality of generated samples in flow matching. By reducing the variance in gradient estimation, enhanced coupling can accelerate training. Additionally, improved coupling could decrease transport costs and straighten trajectories, yielding better-quality samples.

The OT Approach. Prompted by these findings, Pooladian et al. (2023) have explored OT methods to devise batch couplings within the framework of flow matching models. They show that deterministic and non-crossing paths enabled by OT with infinite batch size lowers the variance of gradient estimators. Experimentally, they assess the efficacy of OT solvers, such as Hungarian matching and Sinkhorn algorithms, in coupling batches of noise and data points. Dou et al. (2024) have adopted this approach in consistency models, underscoring the utility of batch OT in boosting model performance. However, a significant challenge remains in the generative modeling of images, where the coupling's effectiveness is limited by the meaningfulness of distances between noise and natural images, and by the batch size.

Another line of works using OT tools with score-based models relies on the Schrodinger Bridge formulation (De Bortoli et al., 2021; Shi et al., 2024; Korotin et al., 2024). However, it has mostly proven benefits on transfer tasks rather than standard generative tasks.

Re-using ODE Couplings. An alternative strategy, termed ReFlow (Liu et al., 2023), leverages existing couplings provided by the ODE in a flow framework. In this section, the ODE is defined as $\mathrm{d}x = v(x_t, t)\,\mathrm{d}t$ with intermediate points calculated as $x_t = tx + (1-t)z$ and $t \in [0, 1]$. Once a neural network v_θ is trained using these ICs, solving the backward ODE defines a deterministic function mapping each sample noise to a sample from the approximated data distribution. Notably, Liu et al. (2023) show that this new coupling can reduce transport costs and straighten trajectories, and can be used iteratively for training models from straighter and straighter trajectories. The authors note however that using this procedure with a PF-ODE solver would not guarantee decreasing transport costs and straightening trajectories. Nonetheless, in this iterative procedure, errors can accumulate due to approximation errors from v_θ and the discretization in the ODE solver.

4. Consistency models with Generator-Induced Coupling (GC)

In this section we introduce our method, named GC, leveraging the generator outputs during training. It consists in using the consistency model itself to re-define the standard independent coupling (IC). We run a series of experiments motivating the use and giving intuition on GC. Notably, we show that the resulting trajectories are closer to the ideal PF-ODE updates than with standard IC.

4.1. Generator-Induced Coupling (GC) Training

The solution proposed in this work, illustrated in Figure 1, consists in leveraging the consistency model itself to build a novel type of coupling. The idea is to leverage the properties and the accumulated knowledge from the consistency model, f_θ, to construct pairs of points. To do so, the first step is to sample an intermediate point, which is done as usual with $x \sim p_\star$, $z \sim p_z$ and an IC between the two distributions, and then predicting its endpoint $\hat{x}$ via the consistency model:

$$(x, z) \sim q_{\mathrm{I}}, \quad x_i = x + \sigma_i z, \quad \hat{x} = \mathrm{sg}(f_\theta(x_i, \sigma_i)). \tag{7}$$

and this $\hat{x}$ is coupled to z, thereby defining our *generator-induced coupling* q, to construct pairs of points $(\tilde{x}_i, \tilde{x}_{i+1})$:

$$(\hat{x}, z) \sim q, \quad \tilde{x}_i = \hat{x} + \sigma_i z, \quad \tilde{x}_{i+1} = \hat{x} + \sigma_{i+1} z. \tag{8}$$

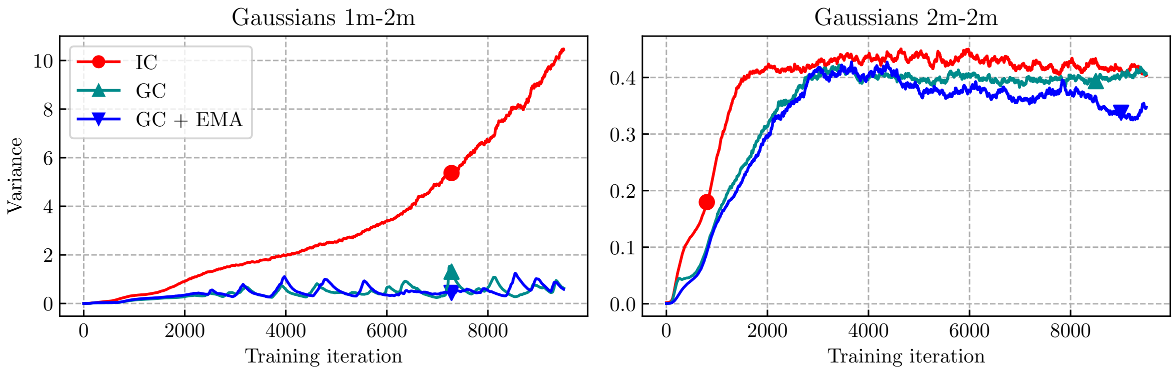


Figure 2. Evolution of the variance of the estimator of the gradient during training of both 1m-2m and 2m-2m synthetic settings. Generator-induced Coupling (GC) training reduces the variance compared to IC training used in the original consistency training algorithm.

Algorithm 1 Training of consistency models with generator-induced trajectories.

```
1:  **Input:** Randomly initialized $f_\theta$, number of timesteps $k$, noise schedule $\sigma_i$, loss weighting $\lambda(\cdot)$, learning rate $\eta$, distance $\mathcal{D}$, noise distribution $p_z$.
2:  **Output:** Trained consistency model $f_\theta$.
3:  **repeat**
4:      $x \sim p_\star$                                   // set of real data points
5:      $z \sim p_z$                                       // set of noise vectors
6:      $i \sim$ multinomial$\big(p(\sigma_0), \ldots, p(\sigma_N)\big)$
7:                                                         // sampling timesteps
8:      $x_i \leftarrow x + \sigma_i z$ // computation of intermediate points
9:      $\hat{x} \leftarrow \mathrm{sg}\big(f_\theta(x_i, \sigma_i)\big)$
                        // endpoint prediction from the consistency model
10:     $\tilde{x}_i \leftarrow \hat{x} + \sigma_i z$, $\tilde{x}_{i+1} \leftarrow \hat{x} + \sigma_{i+1} z$
                        // generator-induced intermediate points
11:     $L(\theta) = \lambda(\sigma_i)\mathcal{D}\big(\mathrm{sg}(f_\theta(\tilde{x}_i, \sigma_i)), f_\theta(\tilde{x}_{i+1}, \sigma_{i+1})\big)$
                        // compute consistency loss
12:     $\theta \leftarrow \theta - \eta \nabla_\theta L(\theta)$          // optimization step
13: **until** convergence
```

The loss function is defined on this generator-induced pair:

$$L_{\text{g-consistency}} = \mathbb{E}_{(\hat{x},z)\sim q, \tilde{x}_i = \hat{x}+\sigma_i z, \tilde{x}_{i+1} = \hat{x}+\sigma_{i+1} z}\Big[\mathcal{D}\Big(\mathrm{sg}(f_\theta(\tilde{x}_i, \sigma_i)), f_\theta(\tilde{x}_{i+1}, \sigma_{i+1})\Big)\Big]. \tag{9}$$

The procedure is presented in Figure 1 and Algorithm 1.

Generator-induced trajectories satisfy the boundary conditions of diffusion processes. Let us note two following important properties of the distribution $p_{\tilde{x}_i}$ of $\tilde{x}_i$: $p_{\tilde{x}_0} = p_0 \approx p_\star$ and $p_{\tilde{x}_N} = p_{\sigma_N} \approx p_z$. The former is achieved thanks to the boundary condition of the consistency model (*c.f.* Section 2.1), and the latter by construction of the GC q preserving the marginal noise distribution.

4.2. Experimental Insights into Model Properties

In this section, we propose to experimentally validate the soundness of our proposed coupling procedure by evaluating key quantities. Sections 2.2 and 3 expose three key properties that could play a role in accelerating the training and improve the performances of consistency models: *(i)* a lower variance of the gradient estimator; *(ii)* a lower data to noise coupling transport cost; and *(iii)* pairs $(\tilde{x}_i, \tilde{x}_{i+1})$ closer to the PF-ODE trajectory, as in the distillation setting, than (x_i, x_{i+1}) in IC.

Experimental settings. To observe whether our proposed method verify these properties in practice, we run experiments on two synthetic settings where the data distribution $p_\star$ is a mixture of 2 Gaussians with equal weight, and the noise distribution p_z is either a single Gaussian or a mixture of 2 Gaussians with equal weight. The goal of the first (resp. second) setting is to map a 1-mode (resp. 2-mode) noise distribution to a 2-mode data distribution; we denote this setting 1m-2m (resp. 2m-2m). The 2m-2m setting, inspired by Liu et al. (2023), facilitates the visualization of intersecting trajectories. We then compare our GC training to the standard IC training.

***(i)* Lower variance of the gradient.** In this experiment, we propose to measure how the variance of the estimator of the gradient of the GC training compares to the one of the standard IC training. As highlighted in Section 3 a lower variance could positively impact the training and performances. We then run a full training, in both 1m-2m and 2m-2m settings for both GC and IC trainings, and log the variance for each batch. As shown in Figure 2, the GC procedure lowers the variance of the gradient's estimator in both settings. Note that, particularly in the 2m-2m setting, using an Exponential Moving Average (EMA) of the parameters of the network on top of our method smoothes the variance specifically at the beginning of the training (around iteration 200) and lowers it as well.

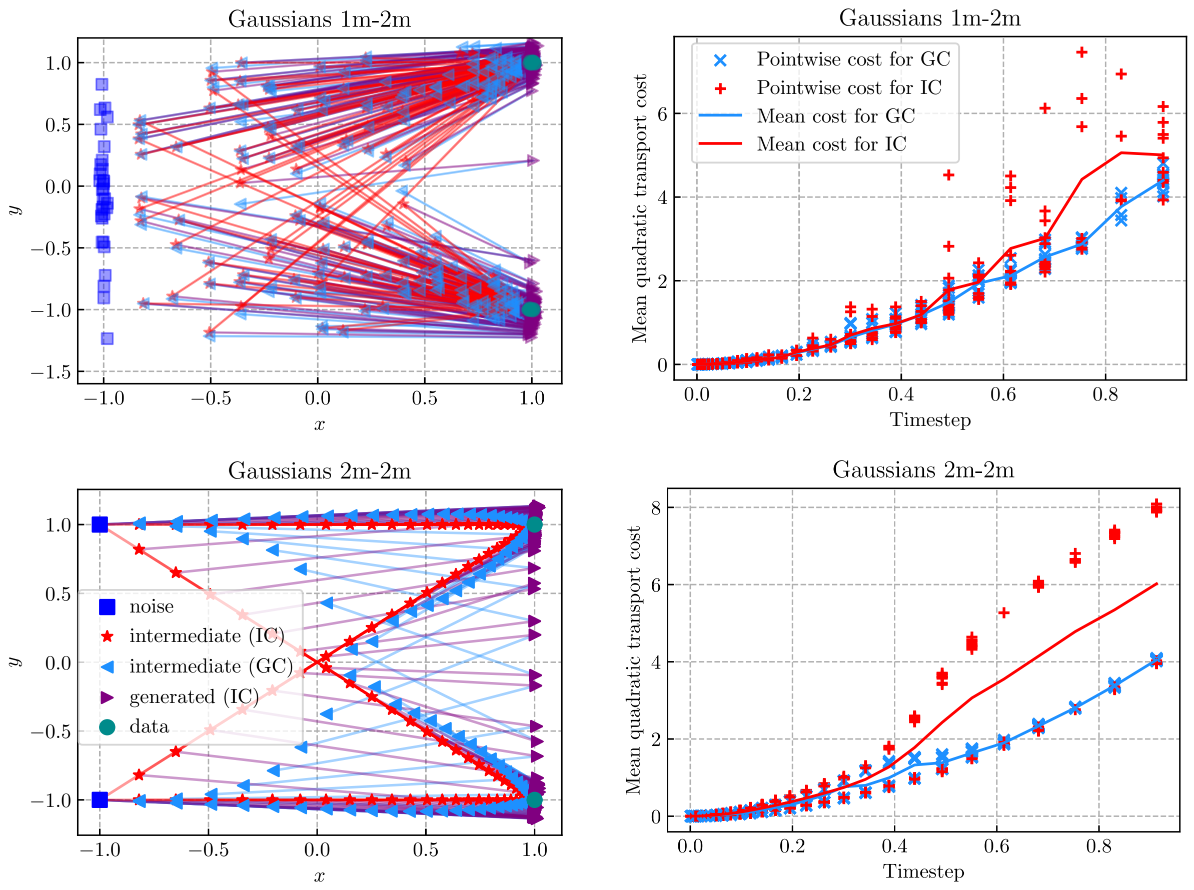


Figure 3. GC and IC trainings on 1m-2m (top) and 2m-2m (bottom). (left) GC samples (◀) are closer to the shortest paths between noise and data distributions than IC (★). (right) Generator-induced trajectories have lower transport cost than standard trajectories. In both cases, transportation cost has two modes.

***(ii)* Lower transport costs.** As stressed in Section 3, a line of works has brought evidence that reducing the data-noise coupling transport cost could fasten the training and help produce better samples. We compare the quadratic transport costs of mapping an intermediate point to its associated true or synthetic data, formally $\mathbb{E}[\|x - x_t\|_2^2]$ and $\mathbb{E}[|\hat{x} - \tilde{x}_t|_2^2]$. The consistency model used for generating $\tilde{x}_t$ has been trained for 2000 steps and has not reached convergence yet. We run this experiment on both 1m-2m and 2m-2m synthetic settings, results are depicted in Figure 3. We observe that $\mathbb{E}[\|x - x_t\|_2^2] \geq \mathbb{E}[\|\hat{x} - \tilde{x}_t\|_2^2]$. Additionally, we observe that GC alters the marginal probabilities, such that $p_{\tilde{x}_i} \neq p_{\sigma_i}$ for $i \in [1, \ldots, N-1]$. Notably, after intersections, the trajectories tend to be attracted to paths between the closest modes. This is confirmed on the right part of the plot where the expected cost is far lower for GC and the two modes in the cost tend to disappear for larger timesteps (denoting that intermediate points $\tilde{x}_t$ are mapped to nearest mode).

***(iii)* Closer to the PF-ODE trajectories.** As discussed in Section 2.2, the goal of consistency models enforcing consistency on pairs of points that belong to a trajectory of the PF-ODE, but pairs of points (x_i, x_{i+1}) built on IC do not satisfy this property. We hypothesize that our $(\tilde{x}_i, \tilde{x}_{i+1})$ built on GC are closer to actual PF-ODE trajectories. To do so, we first learn a score-based model s_Φ in both the 1m-2m synthetic setting and on the CIFAR-10 dataset[1] with an EDM setting. Given s_Φ we can compute a PF-ODE update from any point using Equation (5). We then train consistency models with either standard IC training or GC training. We define true PF-ODE updates $x_i^\Phi = x_{i+1} - (\sigma_i - \sigma_{i+1})\sigma_{i+1}s_\Phi(x_{i+1}, \sigma_{i+1})$ and $\tilde{x}_i^\Phi = \tilde{x}_{i+1} - (\sigma_i - \sigma_{i+1})\sigma_{i+1}s_\Phi(\tilde{x}_{i+1}, \sigma_{i+1})$. Then, we compare their distance with the training pairs, resp. $\mathbb{E}[\|x_i - x_i^\Phi\|_2]$ and $\mathbb{E}[\|\tilde{x}_i - \tilde{x}_i^\Phi\|_2]$. The smaller this quantity, the closer the update to the ideal update obtained from the score-based model. As shown in Figure 4, GC pairs are consistently closer to actual PF-ODE trajectories than IC ones.

[1]Here, the 2m-2m setting would require a bridge process.

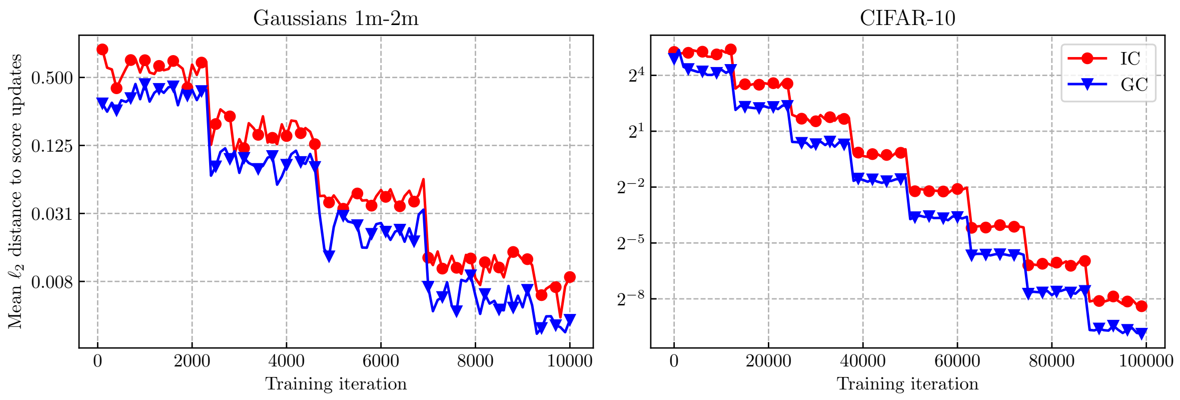


Figure 4. Distance to score-based updates of standard IC pairs (x_i, x_{i+1}) and GC pairs $(\tilde{x}_i, \tilde{x}_{i+1})$ during training. GC pairs are closer to the ideal updates than IC ones.

Table 1. FID on CIFAR-10.

Forward process	Optimizer	FID
EDM	Adam	13.0
EDM	Lion	8.8
Bridge	Lion	**8.0**

5. Experiments on Image Generation

In the previous section we validate experimentally several key properties and visualize the impact of our proposed GC training on two synthetic settings. Here, we experiment GC training to learn consistency models on unconditional image generation tasks without a pre-trained score model.

5.1. Experimental Setting

We apply our approach to three image datasets used in an unconditional setting: CIFAR-10 (Krizhevsky et al.), 64×64 CelebA (Liu et al., 2015), and 32×32 1k-ImageNet (Deng et al., 2009). We compare our model with standard IC consistency training setting with IC and denote this baseline IC. The baseline IC model is built on the training principles and techniques outlined by Song and Dhariwal (2024). Experiments were conducted on NVIDIA A100 40GB GPUs. Our codebase will be publicly released upon publication to ensure reproducibility of the results. Details of our experiments can be found in the Appendix.

To evaluate the contribution of GC w.r.t. the baselines IC and batch-OT, we propose to select a common base model and optimization strategy based on IC that would be used for all models, based on its performances on CIFAR-10 and common practices.

Selecting base components. We start with the standard EDM forward process (Karras et al., 2022; Song et al., 2023; Song and Dhariwal, 2024) defined as $x_i = x + \sigma_i z$, and compare Adam (Kingma and Ba, 2015) to Lion (Chen et al., 2024) with a similar learning rate of $1\mathrm{e}^{-4}$. We train the IC model with both optimizers and observe a significant improvement in FID while using Lion. We then compare EDM and bridge forward processes with Lion. We define the bridge process as $x_i = (1-\alpha_i)x + \alpha_i z$, with $\alpha_i = \frac{\sigma_i}{\sigma_i+1}$ and σ_i taken from the EDM process. In this way, we conserve a similar signal-to-noise ratio at given timestep i. Results are presented in Table 1. We observe a superior performance of the bridge process with Lion, and we thus stick to these two components for the following experiments.

EMA generator-induced trajectories. Another key component of our based model is the EMA of the network parameters for both evaluation and GC-training. Indeed, the following question arises: while training, should GC be computed with an EMA model? Like diffusion and score-based models, previous works (Song and Ermon, 2020) observed that at inference EMA models have steadier and better performance. We thus use EMA for reporting the results of both IC and batch-OT. Additionally, and specifically to GC, we gain some insight from previous section experiments that using EMA during training might reduces the variance of the gradient estimator (see Figure 2). We further report, in Figure 5(b), an ablation study in our final setting which supports the superiority of the EMA.

5.2. Results

In this section we report the results of our proposed method GC and expose our thought process towards the final GC training procedure. Note that we do not achieve the performance of Song and Dhariwal (2024) on CIFAR-10 and 1k-ImageNet due to using smaller models, a smaller batch size (512 vs 1024 on CIFAR-10, 512 vs 4096 on ImageNet), and fewer training steps (100k vs 400k).

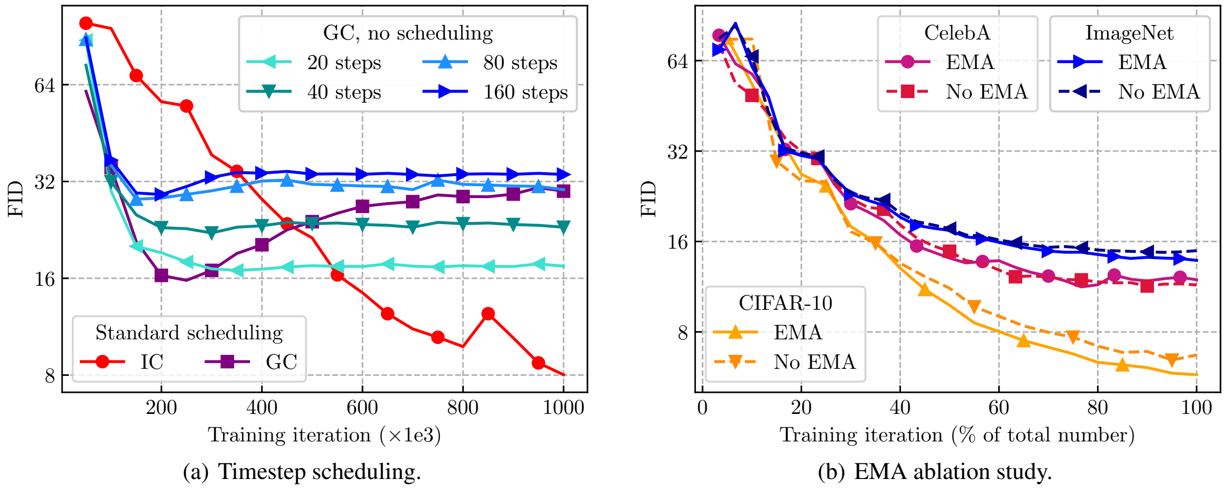


Figure 5. (a) Timestep scheduling comparison for consistency models trained with generator-induced trajectories on CIFAR 10. (b) EMA ablation study in GC training for $\mu = 0.5$.

Timestep scheduling. First, we learn a consistency model using the exact same training procedure as described in Section 4 on CIFAR-10. We compare the performance of GC w.r.t. IC during training using the exponentially increasing number of timesteps from Song and Dhariwal (2024). We report in Figure 5(a) the FID of one-step generated images. We observe that, at the beginning of the training, GC allows for a faster convergence speed than IC. Unfortunately, around the 30k-th iteration, GC suffers from degraded performance. We hypothesize that the timestep scheduling might not be optimal for training consistency models with GC and thus train different GC models with a fixed number of timesteps. Results are reported in Figure 5(a). Interestingly we observe that, with no scheduling, GC offers a faster convergence without performance degradation. Using 20 timesteps, we match the performance of the best GC results obtained with standard scheduling.

Mixing procedure. Note that, even though the GC procedure create new pairs $(\tilde{x}_i, \tilde{x}_{i+1})$, it still needs to apply the consistency model to x_i drawn from IC. Training only on GC can then induce a distribution shift, as illustrated in Figure 3. This can be the reason why we previously observed a degraded performance reported in Figure 5(a). We thus propose a simple yet efficient way to reduce this distribution shift by mixing IC and GC pairs during training, adding only one hyperparameter. We define a mixing factor μ used to mix standard trajectories with generator-induced trajectories. At each training step, training pairs are drawn from GC with probability μ, while the remaining pairs are computed from standard IC. We denote GC ($\mu = \cdot$) the resulting mixing procedure. Thus, GC ($\mu = 0$) corresponds to the standard IC procedure and GC ($\mu = 1$) corresponds to the procedure introduced in Section 4. We run the mixing procedure on the three image dataset and include batch OT (Pooladian et al., 2023; Dou et al., 2024) as a supplementary baseline.

As shown in Figure 6, we observe an interesting interpolation phenomenon between $\mu = 0$ to $\mu = 1$. At $\mu = 0$, we recover the steady FID improvement of IC training. When increasing μ, the convergence of the generative model accelerates. However, when reaching $\mu = 1$, the FID improvement rate is the fastest at the beginning of the training, but then suffers from performance degradation. For $0.3 \leq \mu \leq 0.7$, we observe a sweet spot where the convergence is faster than IC and, for CIFAR-10 and CelebA, the final FID improves compared to both IC and batch-OT baseline models. We show in Figure 7 samples from CelebA 64×64. GC seems to produce sharper and more detailed images specifically when looking at eyes, mouth, and hairs of generated faces.

Wall-clock Training Time. As illustrated in Figure 6, our method converges faster than IC in terms of the number of required training steps. However, each training step is more time-consuming, as it necessitates a forward evaluation of the consistency model without gradient computation. Regarding wall-clock training time on CIFAR-10, 100k training steps under standard conditions require approximately 20 hours on an A100 GPU. In contrast, 100k training steps employing GC extend to about 25 hours. Importantly, despite the increased time per iteration, the hybrid model achieves the minimum Fréchet Inception Distance (FID) sooner than the consistency model, both on CelebA and CIFAR-10, when considering total wall-clock training time and the number of iterations needed for reaching the minimum.

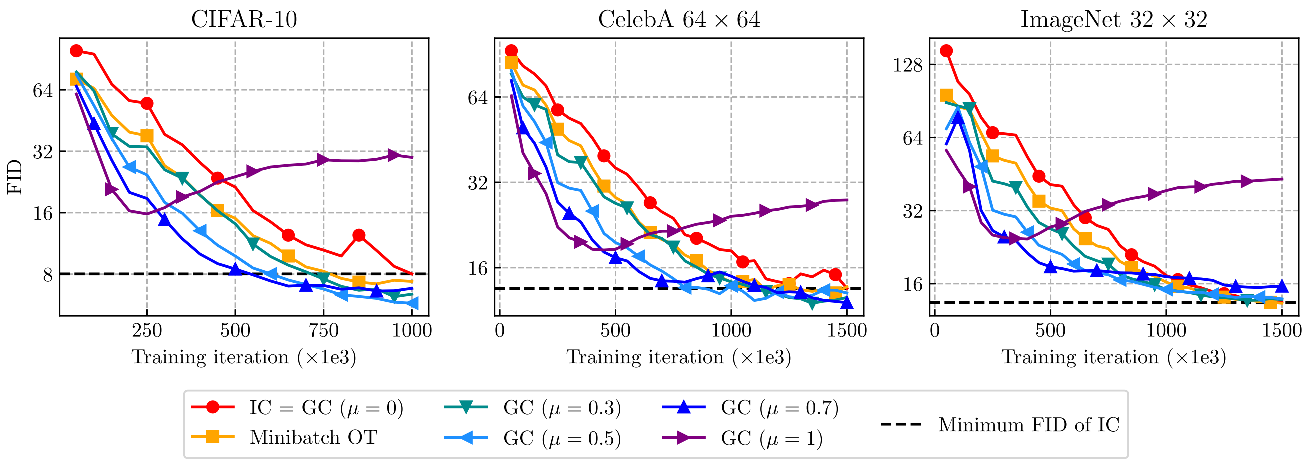
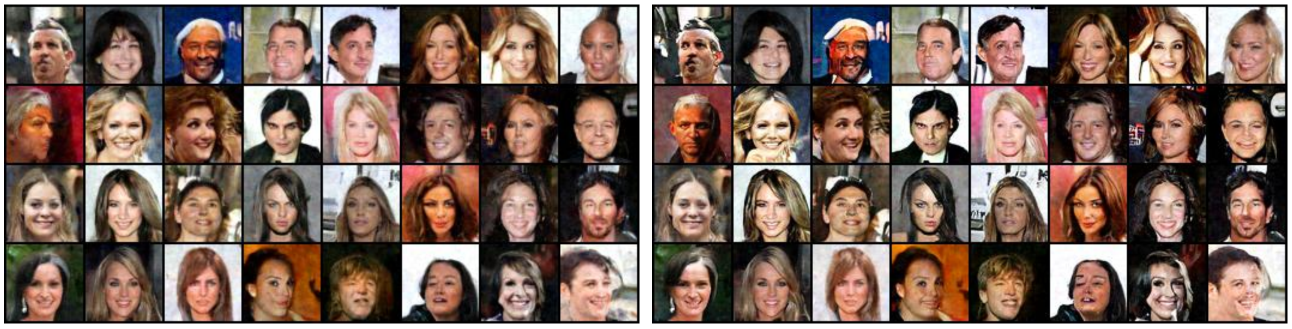


Figure 6. Performance of consistency models using batch-OT or mixed GC ($\mu = \cdot$) for different mixing factors μ. GC benefits from an increased convergence rate over standard IC and achieves better performance than both IC and batch-OT on CIFAR-10 and CelebA for μ equal to 0.5 and 0.7.



(a) IC = GC ($\mu = 0$).

(b) GC ($\mu = 0.5$).

Figure 7. Uncurated samples from consistency models trained on CelebA 64 × 64 for fixed noise vectors. Note that models trained with generator-induced trajectories tend to generate sharper images.

6. Conclusion

In this paper, we introduce a generator-induced coupling and new trajectories for training consistency models without a pre-trained score model. We experimentally give intuition on the interest of those trajectories. Most notably, they are closer to ideal trajectories from a pre-trained score update than standard ones. This results in a faster convergence in terms of both the number of training steps and wall-clock training time. Our approach paves the way for a new type of coupling induced by the generator itself. Interestingly, our method offers a fresh viewpoint on this problem, which is usually tackled with traditional OT tools.

Limitations. Training consistency models with generator-induced trajectories leads to intriguing behaviors that we still do not fully understand. For example, why do they exhibit different behavior than standard consistency models regarding the scheduling of timesteps during training? This is surprising since Song et al. (2023) showed that increasing the number of timesteps leads to a decreased approximation error of the consistency model. Future work should carefully re-examine hyperparameter choices for consistency models trained with generator-induced trajectories, especially for scheduling. Another technical limitations of our work is, at this time, the lack of theoretical support on the benefits of generator-induced trajectories. The advantages they bring are evaluated experimentally on synthetic settings and computer vision datasets, but then limited to these scopes.

References

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in Neural Information*

Processing Systems, 2024.

Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009.

Hongkun Dou, Junzhe Lu, Jinyang Du, Chengwei Fu, Wen Yao, Xiao qian Chen, and Yue Deng. A unified framework for consistency generative modeling, 2024. URL https://openreview.net/forum?id=Qfqb8ueIdy.

Don Fallis. The epistemic threat of deepfakes. *Philosophy & Technology*, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33, 2020.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35, 2022.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation. *Advances in Neural Information Processing Systems*, 36, 2024.

Alexander Korotin, Nikita Gushchin, and Evgeny Burnaev. Light schrödinger bridge. In *The Twelfth International Conference on Learning Representations*, 2024.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky TQ Chen. Multisample flow matching: Straightening flows with minibatch couplings. In *International Conference on Machine Learning*, 2023.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.

Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. Diffusion schrödinger bridge matching. *Advances in Neural Information Processing Systems*, 2024.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *The Twelfth International Conference on Learning Representations*, 2024.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 11918–11930. Curran Associates, Inc., 2019.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33, 2020.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23 (7), 2011.

A. Appendix

A.1. Broader Impacts

If used in large-scale generative models, notably in text-to-image models, this work may increase potential negative impacts of deep generative models such as *deepfakes* (Fallis, 2020).

A.2. Experimental Details

Scheduling functions and hyperparameters from Song and Dhariwal (2024). The training of consistency models heavily rely on several scheduling functions. First, there is a noise schedule $\{\sigma_i\}_{i=0}^{N}$ which is chosen as in Karras et al. (2022). Precisely, $\sigma_i = \left({\sigma_0}^{\frac{1}{\rho}} + \frac{i}{N}({\sigma_\text{N}}^{\frac{1}{\rho}} - {\sigma_0}^{\frac{1}{\rho}})\right)^{\rho}$ with $\rho = 7$. Second, there is a weighting function that affects the training loss. It is chosen as $\lambda(\sigma_i) = \frac{1}{\sigma_{i+1} - \sigma_i}$. Combined with the choice of noise schedule, it emphasizes to be consistent on timesteps with low noise. Then, Song et al. (2023) proposed to progressively increase the number of timesteps N during training. In their most recent work, they argue that a good choice of dicretization schedule is an exponential one, which gives $N(k) = \min(s_0 2^{\lfloor \frac{k}{K'} \rfloor}, s_1) + 1$ where $K' = \lfloor \frac{K}{\log_2[s_1/s_0]+1} \rfloor$, K is the total number of training steps, k is the current training step, s_0 (respectively s_1) the initial (respectively final) number of timesteps. Finally, Song and Dhariwal (2024) propose a discrete probability distribution on the timesteps which mimics the continuous probability distribution recommended in the continuous training of score-based models by Karras et al. (2022). It is defined as $p(\sigma_i) \propto \text{erf}(\frac{\log(\sigma_{i+1}) - P_\text{mean}}{\sqrt{2}P_\text{std}}) - \text{erf}(\frac{\log(\sigma_i) - P_\text{mean}}{\sqrt{2}P_\text{std}})$.

In practice, Song and Dhariwal (2024) recommend using: $s_0 = 10$, $s_1 = 1280$, $\rho = 7$, $P_\text{mean} = -1.1$, $P_\text{std} = 2.0$.

Details on bridge process. For the bridge process, note that we first sample σ_i from $\sigma_i = \left({\sigma_0}^{\frac{1}{\rho}} + \frac{i}{N}({\sigma_\text{N}}^{\frac{1}{\rho}} - {\sigma_0}^{\frac{1}{\rho}})\right)^{\rho}$.

On Gaussians experiment, we use $\rho = 3$, $\sigma_0 = 0.001$ and $\sigma_\text{N} = 1$. Then, we compute $\alpha_i = \sigma_i$ and define intermediate points such as $x_i = \alpha_i x + (1 - \alpha_i)z$.

On image experiments, we use $\rho = 7$, $\sigma_0 = 0.001$ and $\sigma_\text{N} = 80$. Then, we compute $\alpha_i = \frac{\sigma_i}{\sigma_i + 1}$ and define intermediate points such as $x_i = \alpha_i x + (1 - \alpha_i)z$.

Details on neural networks architectures. On Gaussians, we use simple MLPs with GELU activation functions. On image datasets, we use the NCSN++ architecture (Song et al., 2021) and follow the implementation from https://github.com/NVlabs/edm.

Table 2. Hyperparameters for the Gaussians experiments.

Hyperparameter	Value
number of samples	10 000
batch size	512
training steps	10 000
learning rate	0.000 05
s_0	30
s_1	30
ρ	3
σ_0	0.001
σ_1	1
model	MLP
depth	4
hidden dim	256

Details on computational ressources As mentioned in the paper, the image dataset experiments have been conducted on some NVIDIA A100 40GB GPUs. The Gaussians experiments have also been computed on GPUs (V100 or A100) and run in few minutes.

Table 3. Hyperparameters for CIFAR-10. Arrays indicate quantities per resolution of the UNet model.

Hyperparameter	**Value**
batch size	512
image resolution	32
training steps	100 000
learning rate	0.0001
optimizer	lion
s_0	10
s_1	1280
ρ	7
σ_0	0.001
σ_1	80
network architecture	SongUNet
	(from Karras et al. (2022) implementation)
model channels	128
dropout	0.3
num blocks	3
embedding type	positional
channel multiplicative factor	$[1, 2, 2]$
attn resolutions	$\emptyset$

Table 4. Hyperparameters for CelebA. Arrays indicate quantities per resolution of the UNet model. All models are trained with two learning rates and the best is selected.

Hyperparameter	**Value**
batch size	128
image resolution	64
training steps	150 000
learning rate	$\{0.0001, 0.000\,05\}$
optimizer	lion
s_0	10
s_1	1280
ρ	7
σ_0	0.001
σ_1	80
network architecture	SongUNet
	(from Karras et al. (2022) implementation)
model channels	128
dropout	$[0.05, 0.05, 0.1, 0.2]$
num blocks	$[2, 3, 4, 5]$
embedding type	positional
channel multiplicative factor	$[1, 2, 2, 2]$
attn resolutions	$\emptyset$

Table 5. Hyperparameters for ImageNet. Arrays indicate quantities per resolution of the UNet model. All models are trained with two learning rates and the best is selected.

Hyperparameter	**Value**
batch size	512
image resolution	32
training steps	$150\,000$
learning rate	$\{0.0001, 0.000\,05\}$
optimizer	lion
s_0	10
s_1	1280
ρ	7
σ_0	0.001
σ_1	80
network architecture	SongUNet (from Karras et al. (2022) implementation)
model channels	128
dropout	$[0.1, 0.3, 0.3]$
num blocks	$[4, 5, 7]$
embedding type	positional
channel mult	$[1, 1, 2]$
attn resolutions	16